

GPU for Accelerator Physics at ALS

H. Nishimura, K. Song, Y. Qin

Monthly AFRD Modeling & Simulation Meeting, May 21, 2013



Agenda

- Introduction (5 min), H. Nishimura
- GPU for Particle Tracking (10 min), K. Song
- GPU for Undulator Calculation (20min), Y. Qin



Members

- ALS Acc Phys (D. Robin)
 - H. Nishimura, C. Sun, H. Tarawneh
- IT HPC (G. Jung)
 - Y. Qin, K. Song, K. Muriki, S. Jones
- Hiroshima U.
 - S. Sasaki, A. Miyamoto



GPU at ALS

- The Beamline Scientists adopted it
 - A. Hexemer, P. Zwart, .. are the pioneers at ALS
- Later, Acc. Physics followed them



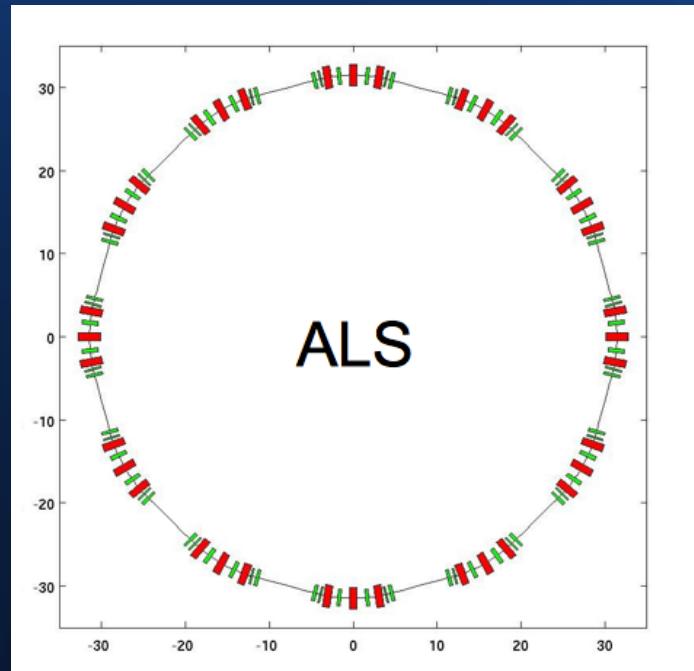
Two Projects

- Dynamic Aperture Calculation
 - Tracy
 - C/C++ → CUDA
 - Kai Song
- Undulator Radiation Calculation
 - Prof. Sasaki's code in Fortran
 - A. Miyamoto rewrote it in C#
 - Then, C/C++ → CUDA
 - Yong Qin



Dynamic Aperture Calculation Background

Multi-variable, multi-objective optimization problem



Adjust the
magnet knobs

Achieve better
performance

Multi-Variables:
- quadrupole
- sextupole

Multi-objectives:
- emittance
- beta function
- dynamic aperture
- beam lifetime



Dynamic Aperture Calculation Background

Genetic Algorithm Implementation

- Use MPI to establish a Master/Slave model

Master: performs genetic operations

- Generate population
- Let slave to produce children out of the population (in parallel)
- Cross-over, mutate and sort the mixed population
- Evaluate and select the population to produce the next generation

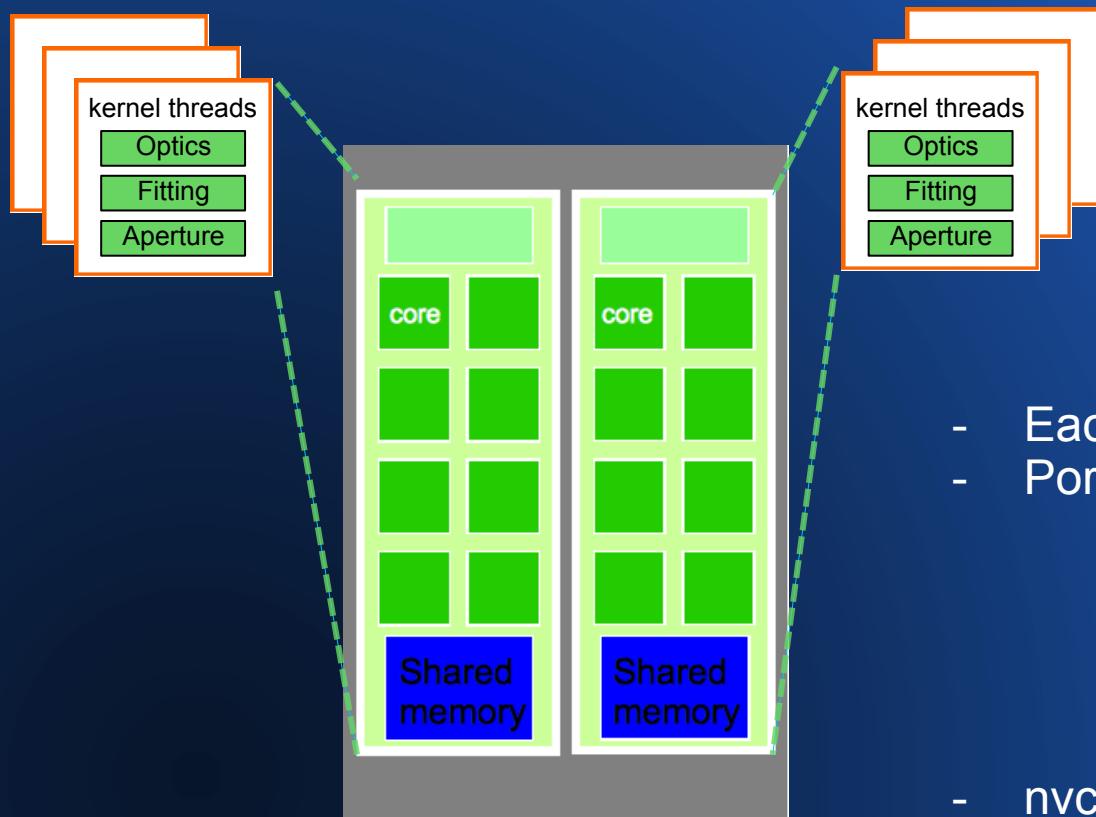
Slave:

- evaluate the lattice properties for a given child
 - Optics
 - Fitting
 - Aperture
- send the result back to the master



GPU Adaptation

Initial Approach

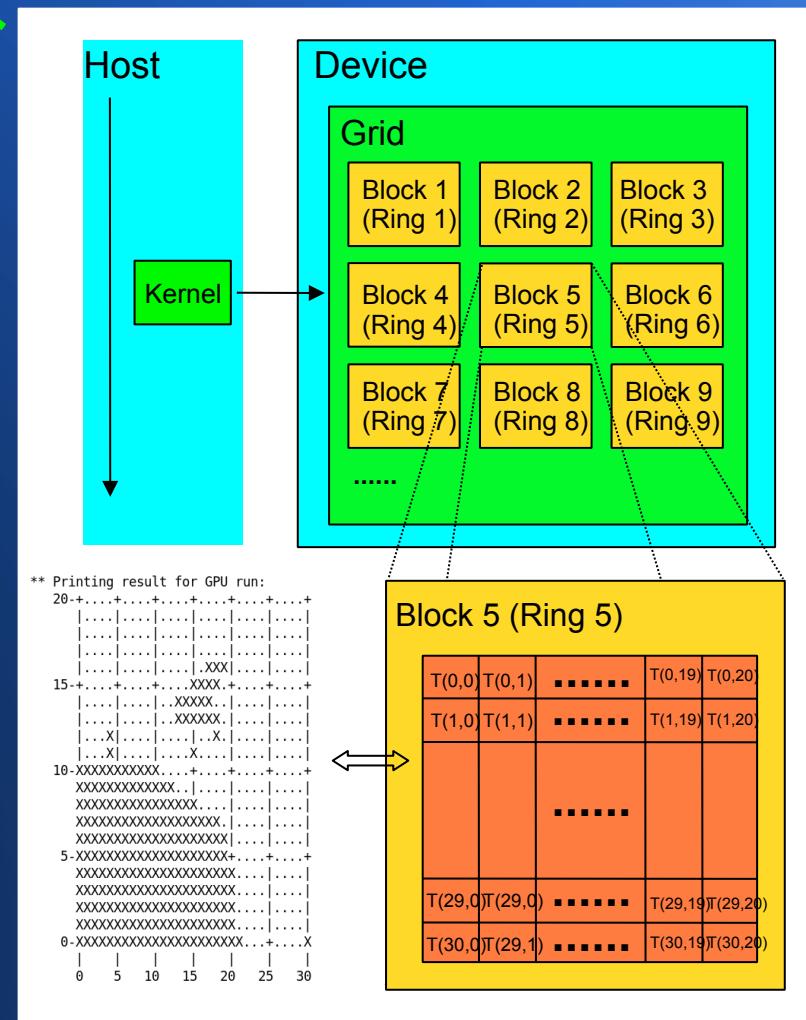
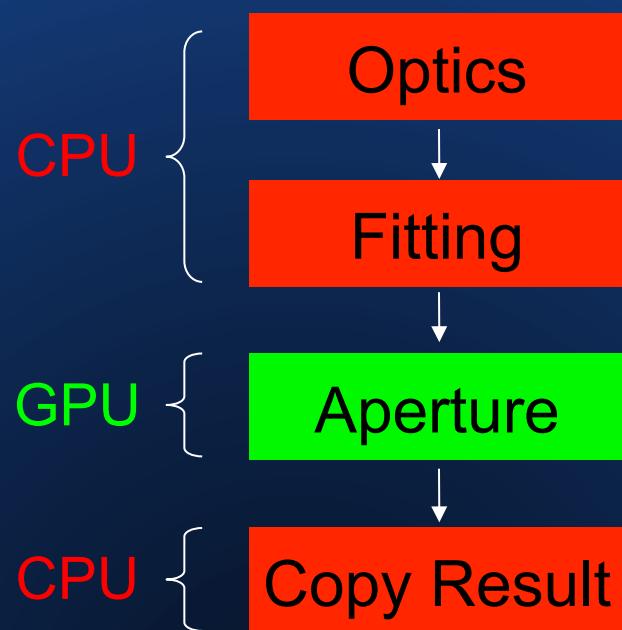


- Each GPU thread takes a MPI process
- Porting requires two major work:
 1. simplify and rewrite data structure of Tracy++ library to fit in GPU
 2. implement slave process components: Optics, Fitting, and Aperture
- nvcc compiler limitation



GPU Adaptation

Implementation



GPU Adaptation

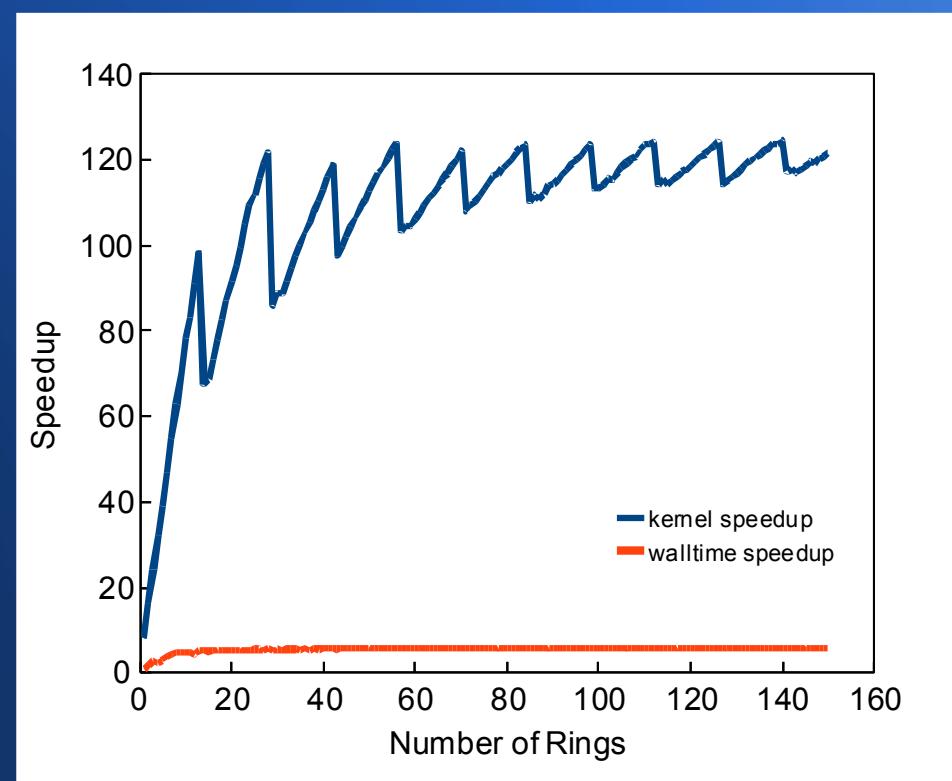
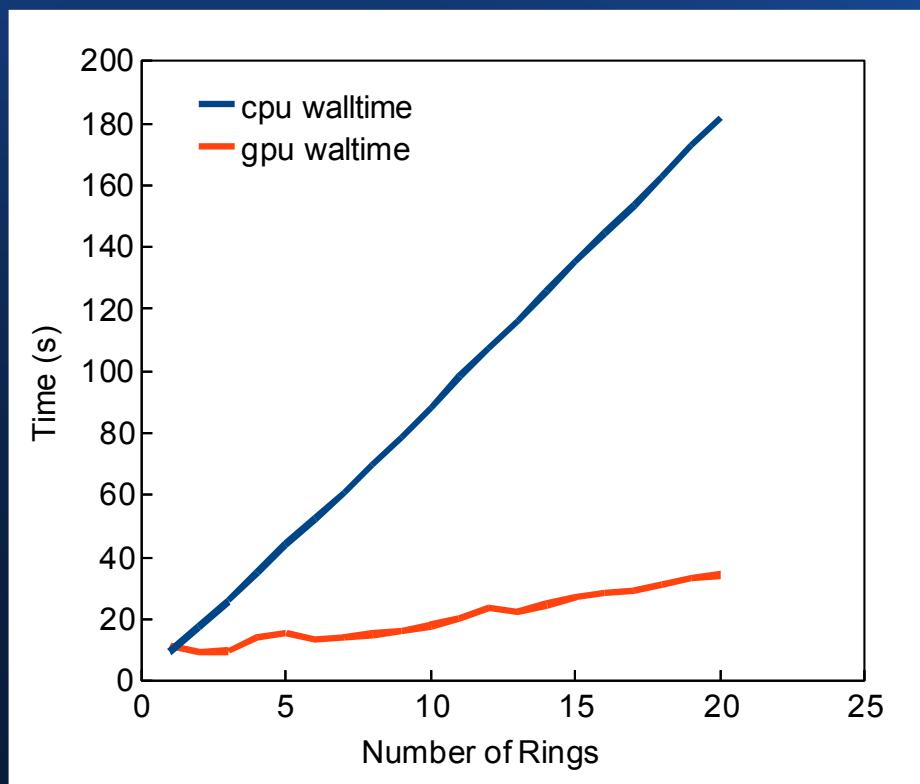
GPU architecture:

- Tesla C2050, Fermi
- 14 multiprocessor
- 32 cores/MP
- total of 448 cores
- threads need to be small and compact
- Cache and Shared Memory Size: 64KB



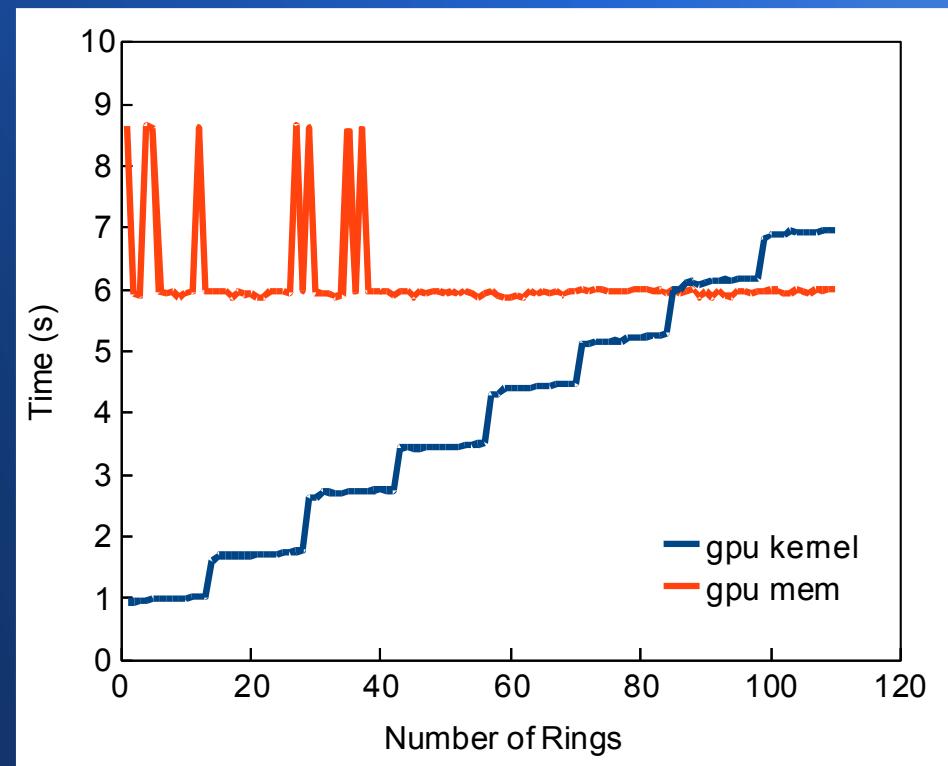
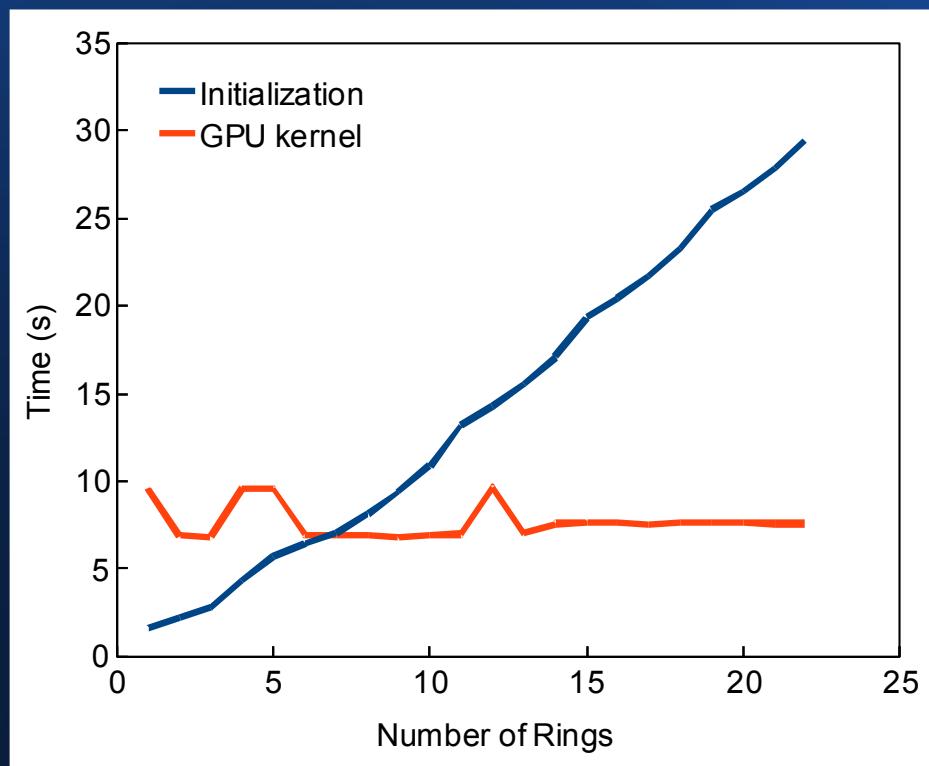
GPU Adaptation

Results (walltime performance)



GPU Adaptation

Results (GPU kernel)



GPU Adaptation

Lessons Learned

Porting existing C/C++ library to GPU is not trivial

- dealing with data structures with many pointers/classes
- without having GPU architecture in consideration
- memory limitation of GPU kernel

Re-evaluate the original data structure and algorithm

- due to architectures difference between GPU and CPU
- Somebody with a comprehensive insight of the library is extremely helpful
- optimize specifically for GPU



Cloud GPU

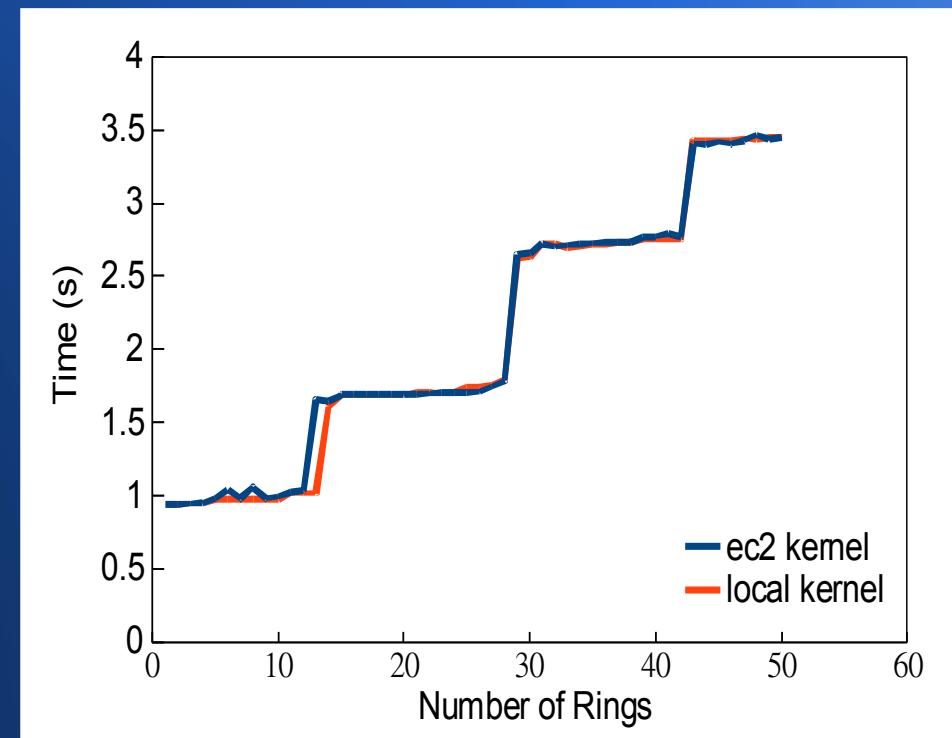
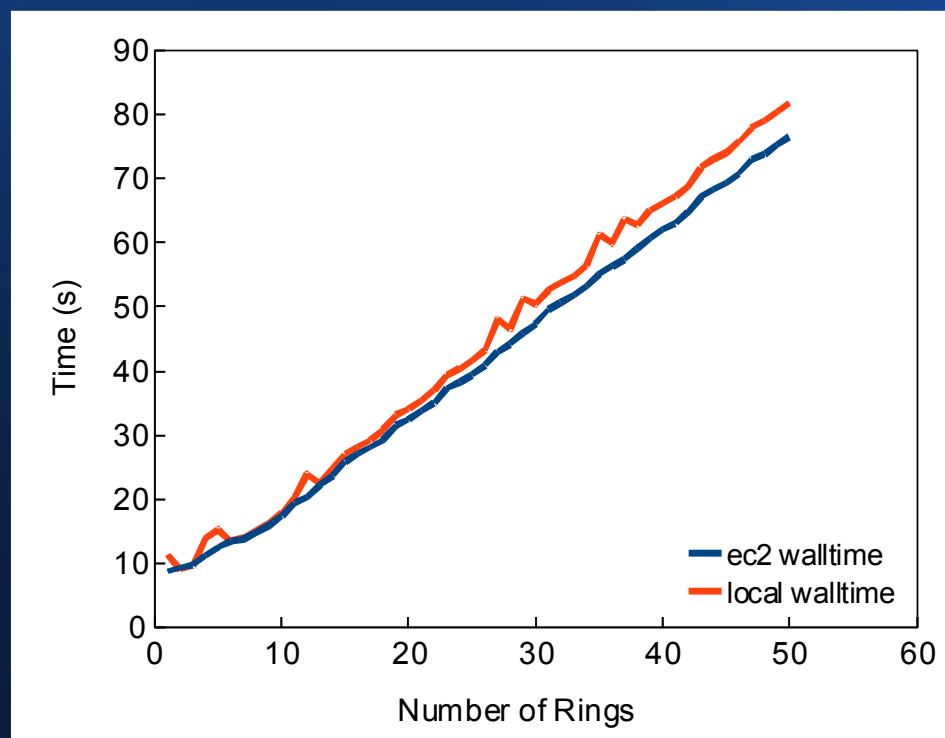
EC2 Cluster GPU Instance

- CPU architecture same as Cluster Compute Instance
- 2 x Nvidia Tesla “Fermi” M2050 GPUs
- 22GB memory
- API name: cg1.4xlarge
- On demand \$2.10/instance-hour
- Not available for Windows OS
- Very close to local GPU used



Cloud GPU

Comparison



Conclusion

A case study using Lattice Optimization Application

- CPU intense, low data processing and low communication

Porting to GPU

- Porting existing C/C++ code is not trivial
- Understand data structure and algorithm
- Identify the suitable part to be parallelized

EC2 Cloud GPU

- Performance is almost identical to local GPU
- Good alternative for local GPU



Publications

H. Nishimura *et al.*, “GPU Computing for Particle Tracking”,
Proceedings of PAC 2011, New York, WEP150.
<http://accelconf.web.cern.ch/accelconf/pac2011/papers/wep150.pdf>

C. Sun *et al.*, “HPC Cloud Applied to Lattice Optimization”,
Proceedings of PAC 2011, New York, WEP151.
<http://accelconf.web.cern.ch/accelconf/pac2011/papers/wep151.pdf>

C. Sun *et al.*, “Low-Emissance Lattice Desgins For ALS Ultimate Upgrade”, Proceedings of PAC 2011, New York, WEP031.
<http://accelconf.web.cern.ch/accelconf/pac2011/papers/wep031.pdf>



Thank You

Questions?

